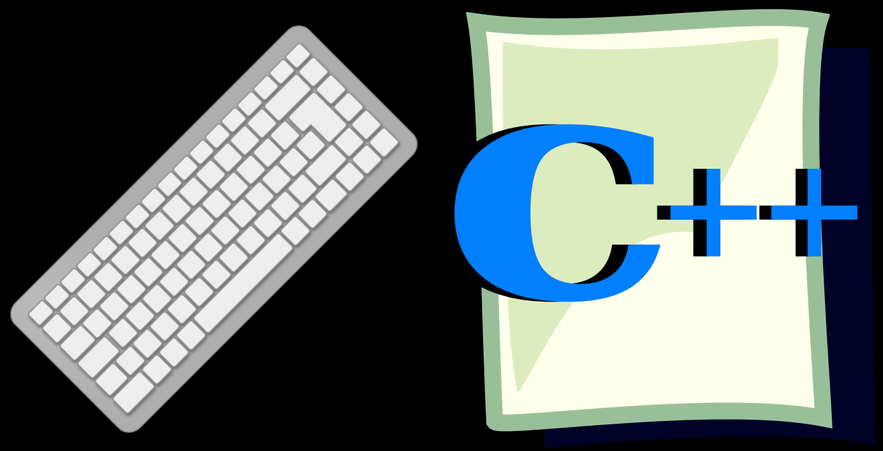
UNIVERSIDAD TECNOLOGICA NACIONAL
Facultad Regional Bahía Blanca

Instrucciones de entrada / salida



Lectura de datos desde el teclado

Valores proporcionados por el usuario

`cin` (`iostream`) — *`c`haracter `in`put stream*

Lectura de valores de variables: operador >> (*extractor*)

```
cin >> cantidad;
```

`cin` `>> cantidad;`

Memoria

cantidad	12


```
1 2
_
```

Valores proporcionados por el usuario

El extractor >>

```
cin >> variable;
```

Transforma los caracteres introducidos en datos

Cursor parpadeante: lugar de lectura del siguiente carácter

La entrada termina con Intro (cursor a la siguiente línea)

¡El destino del extractor debe ser SIEMPRE una variable!

Se ignoran los espacios en blanco iniciales

Valores proporcionados por el usuario

Lectura de valores enteros (`int`)

Se leen dígitos hasta encontrar un carácter que no lo sea

`12abc↲` `12 abc↲` `12 abc↲` `12↲`

Se asigna el valor 12 a la variable

El resto queda pendiente para la siguiente lectura

Recomendación: Lee cada variable en una línea `12↲`

Lectura de valores reales (`double`)

Se leen dígitos, el punto decimal y otros dígitos

`39.95.5abc↲` `39.95 abc↲` `39.95↲`

Se asigna el valor 39,95 a la variable; el resto queda pendiente

Recomendación: Lee cada variable en una línea `39.95↲`

Valores proporcionados por el usuario

¿Qué pasa si el usuario se equivoca?

El dato no será correcto

Aplicación profesional: código de comprobación y ayuda

Aquí supondremos que los usuarios no se equivocan

En ocasiones añadiremos comprobaciones sencillas

Para evitar errores, lee cada dato en una instrucción aparte

Valores proporcionados por el usuario

¿Qué pasa si el usuario se equivoca?

```
int cantidad;
double precio, total;
cout << "Introduce la cantidad: ";
cin >> cantidad;
cout << "Introduce el precio: ";
cin >> precio;
cout << "Cantidad: " << cantidad << endl;
cout << "Precio: " << precio << endl;
```

¡Amigable con el usuario!
¿Qué tiene que introducir?

```
Introduce la cantidad: abc
Introduce el precio: Cantidad: 0
Precio: 1.79174e-307
```

No se puede leer un entero → 0 para `cantidad` y Error
La lectura del precio falla: `precio` no toma valor (*basura*)

Valores proporcionados por el usuario

¿Qué pasa si el usuario se equivoca?

```
Introduce la cantidad: 12abc
Introduce el precio: Cantidad: 12
Precio: 0
```

12 para `cantidad`
No se puede leer un real
→ 0 para `precio` y Error

```
Introduce la cantidad: 12.5abc
Introduce el precio: Cantidad: 12
Precio: 0.5
```

12 para `cantidad`
`.5` → 0,5 para `precio`
Lo demás queda pendiente

```
Introduce la cantidad: 12
Introduce el precio: 39.95
Cantidad: 12
Precio: 39.95
```

¡¡¡Lectura correcta!!!

Programa con lectura de datos

División de dos números

Pedir al usuario dos números y mostrarle el resultado de dividir el primero entre el segundo

Algoritmo.-

Datos / cálculos

1. Pedir el `numerador`

Variable `numerador` (`double`)

2. Pedir el `denominador`

Variable `denominador` (`double`)

3. Realizar la división, guardando el `resultado`

Variable `resultado` (`double`)

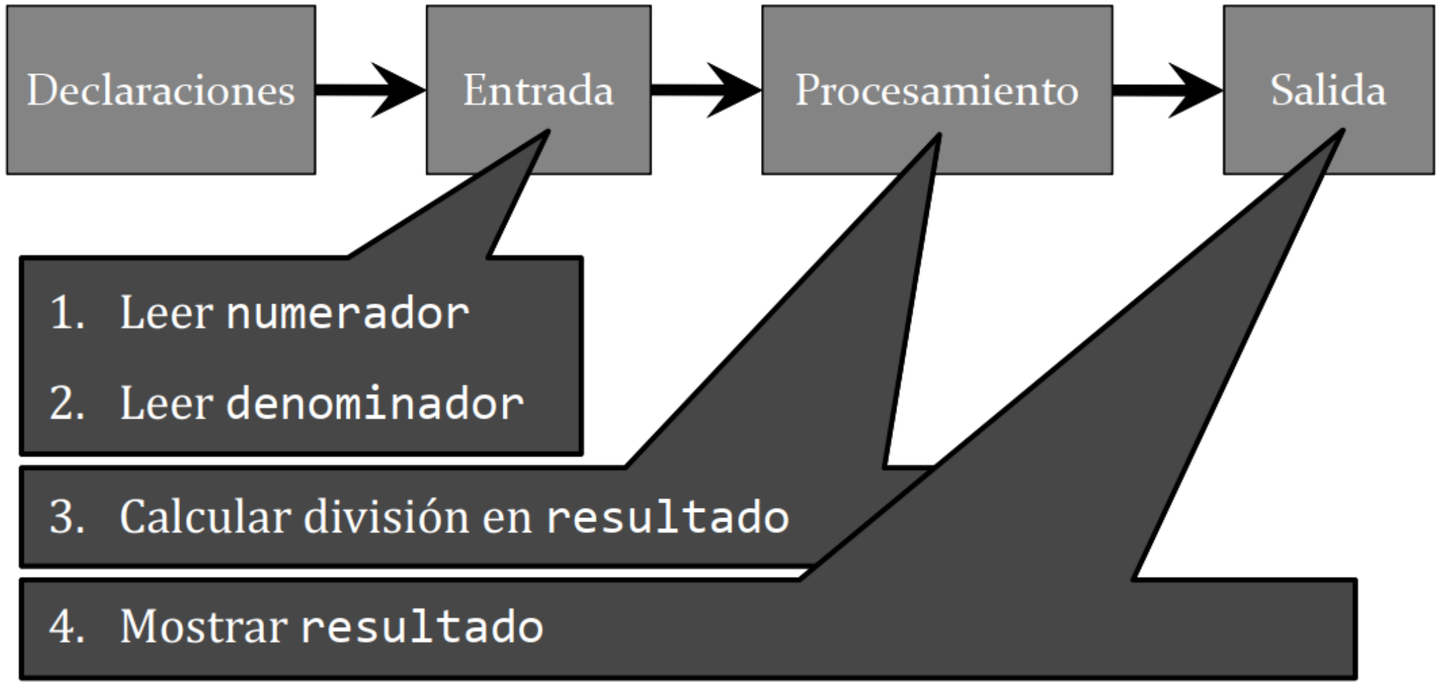
```
resultado = numerador / denominador
```

4. Mostrar el `resultado`

Un esquema general

Entrada-Proceso-Salida

Muchos programas se ajustan a un sencillo esquema:



Programa con lectura de datos

Instrucciones

División de dos números

Pedir al usuario dos números y mostrarle el resultado de dividir el primero entre el segundo.

1. Leer `numerador`

```
cin >> numerador;
```

2. Leer `denominador`

```
cin >> denominador;
```

3. Calcular división en `resultado`

```
resultado = numerador / denominador;
```

4. Mostrar `resultado`

```
cout << resultado;
```

Programa con lectura de datos

Implementación

División de dos números

división.cpp

```
#include <iostream>
using namespace std;

int main()
{
```

Declaraciones	`double numerador, denominador, resultado;`

Entrada	`cout << "Numerador: ";`
	`cin >> numerador;`
	`cout << "Denominador: ";`
	`cin >> denominador;`

Procesamiento	`resultado = numerador / denominador;`
Salida	`cout << "Resultado: " << resultado << endl;`

```
    return 0;
}
```

```
            129
Denominador: 2
Resultado: 64.5
```

Entrada/salida por consola

Entrada/salida por consola (teclado/pantalla)

Flujos de texto (streams)

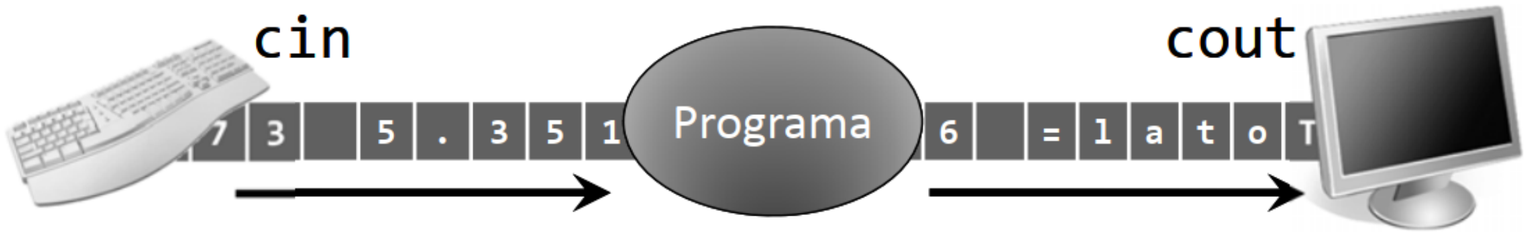
```
#include <iostream>
using namespace std;
```

Conectan la ejecución del programa con los dispositivos de E/S

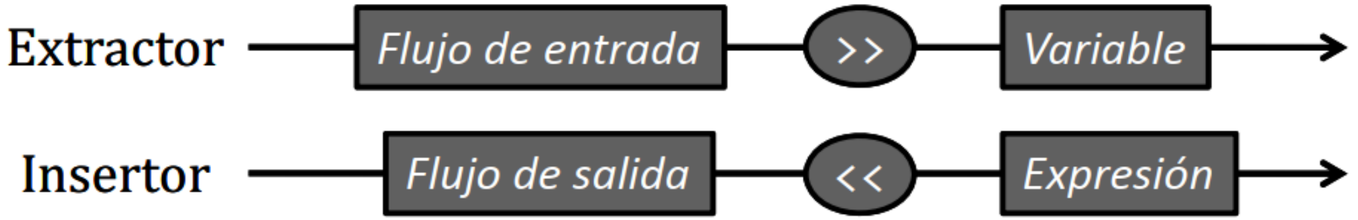
Son secuencias de caracteres

Entrada por teclado: flujo de entrada `cin` (tipo `istream`)

Salida por pantalla: flujo de salida `cout` (tipo `ostream`)



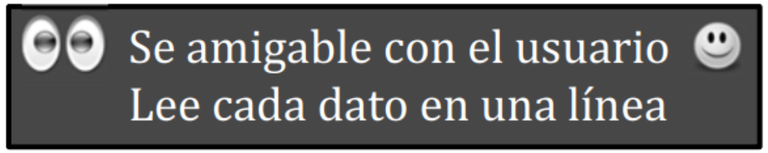
Biblioteca `iostream` con espacio de nombres `std`



Entrada por teclado

Salta los *espacios en blanco* (espacios, tabuladores o saltos de línea)

- `char`
  Se lee un carácter en la variable
- `int`
  Se leen dígitos y se transforman en el valor a asignar
- `float/double`:
  Se leen dígitos (quizá el punto y más dígitos) y se asigna el valor
- `bool`:
  Si se lee `1`, se asigna `true`; con cualquier otro valor se asigna `false`




```
cout << "Introduce tu edad: ";
cin >> edad;
```

Lectura de cadenas (string)

```
#include <string>
using namespace std;
```

`cin >> cadena` termina con el primer espacio en blanco

`cin.sync()` descarta la entrada pendiente

```
string nombre, apellidos;
cout << "Nombre: ";
cin >> nombre;
cout << "Apellidos: ";
cin >> apellidos;
cout << "Nombre completo: "
     << nombre << " "
     << apellidos << endl;
```

```
Nombre: Luis Antonio
Apellidos: Nombre completo: Luis Antonio
```

apellidos recibe "Antonio"

```
string nombre, apellidos;
cout << "Nombre: ";
cin >> nombre;
cin.sync();
cout << "Apellidos: ";
cin >> apellidos;
cout << ...
```

```
Nombre: Luis Antonio
Apellidos: Hernández Yáñez
Nombre completo: Luis Hernández
```

¿Cómo leer varias palabras?

Entrada por teclado

Lectura sin saltar los espacios en blanco iniciales

Llamada a funciones con el operador punto (`.`) :

El operador punto permite llamar a una función sobre una variable

`variable.función(argumentos)`

Lectura de un carácter sin saltar espacios en blanco:

```
cin.get(c); // Lee el siguiente carácter
```

Lectura de cadenas sin saltar los espacios en blanco:

```
getline(cin, cad);
```

Lee todo lo que haya hasta el final de la línea (Intro)

Recuerda:

Espacios en blanco son espacios, tabuladores, saltos de línea, ...

Salida por pantalla

Representación textual de los datos

```
int meses = 7;
cout << "Total: " << 123.45 << endl << " Meses: " << meses;
```

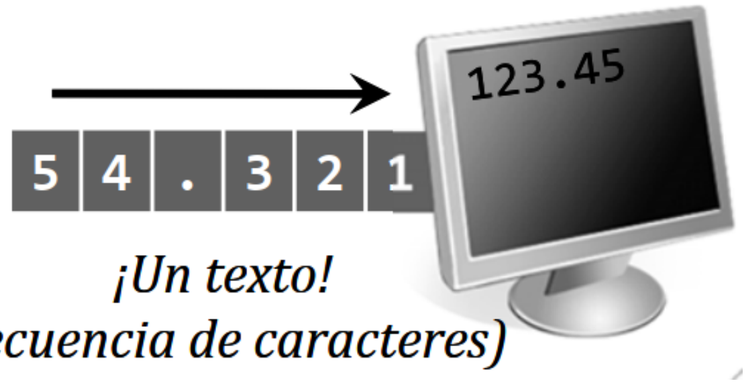
El valor `double` `123.45` se guarda en memoria en binario

Su representación textual es: `'1' '2' '3' '.' '4' '5'`

```
double d = 123.45;
```

d 123.45 *¡Un número real!*

```
cout << d;
```



¡Un texto!
(secuencia de caracteres)

La biblioteca `iostream` define la constante `endl` como un salto de línea

Salida por pantalla


```
int meses = 7;
cout << "Total: " << 123.45 << endl << " Meses: " << meses;
```

```
Total: 123.45
Meses: 7
```

Formato de la salida

`#include <iomanip>`

Constantes y funciones a enviar a `cout` para ajustar el formato de salida

Biblioteca	Constante/función	Propósito
`iostream`	`showpoint / noshowpoint`	Mostrar o no el punto decimal para reales sin decimales (`34.0`)
	`fixed`	Notación de punto fijo (reales) (`123.5`)
	`scientific`	Notación científica (reales) (`1.235E+2`)
	`boolalpha`	Valores `bool` como `true` / `false`
	`left / right`	Ajustar a la izquierda/derecha (por defecto)
`iomanip`	`setw(`*anchura*`)`*	Nº de caracteres (anchura) para el dato
	`setprecision(`*p*`)`	Precisión: Nº de dígitos (en total) Con `fixed` o `scientific`, nº de decimales

*`setw()` sólo afecta al siguiente dato que se escriba, mientras que los otros afectan a todos

Formato de la salida

```
bool fin = false;
cout << fin << "->" << boolalpha << fin << endl;
double d = 123.45;
char c = 'x';
int i = 62;
cout << d << c << i << endl;
cout << "|" << setw(8) << d << "|" << endl;
cout << "|" << left << setw(8) << d << "|" << endl;
cout << "|" << setw(4) << c << "|" << endl;
cout << "|" << right << setw(5) << i << "|" << endl;
double e = 96;
cout << e << " - " << showpoint << e << endl;
cout << scientific << d << endl;
cout << fixed << setprecision(8) << d << endl;
```

```
0->false


123.45x62
|  123.45|
|123.45  |
|x   |
|   62|

96 - 96.0000
1.234500e+002
123.45000000
```